Documentation for maxwell.py

This Python script simulates electromagnetic wave propagation using finite difference time domain (FDTD) methods. It utilizes the NumPy library for numerical operations and Matplotlib for visualizing the results through animations.

Overview of the Code

The script initializes physical constants and simulation parameters, sets up the fields for electromagnetic waves, and iteratively computes the evolution of these fields over a specified number of timesteps. The results are displayed using an animated heatmap representation of the electric field.

Code Structure

Variables

- `num_timesteps` : Number of timesteps for the simulation, set to 1200.
- `n` : Size of the grid (1000 x 1000) for the simulation.
- `epsilon` : Permittivity of free space, a physical constant.
- `mew` : Permeability of free space, another physical constant.
- `Ez` : 2D NumPy array to hold the electric field (E) values along the z-axis.
- `Hy` : 2D NumPy array to hold the magnetic field (H) values along the y-axis.
- `Hx` : 2D NumPy array to hold the magnetic field (H) values along the x-axis.
- `c` : Calculated speed of light in the medium.
- `S` : Stability factor, set to 1/sqrt(2).
- `dx` : Spatial step size, set to 1 micron.
- `dt` : Time step size calculated based on stability criteria.
- `lambd` : Wavelength of the wave, defined as 20 times the spatial step size.
- `omega` : Angular frequency calculated using the speed of light and wavelength.
- `Ca` , `Da` : Constants used in the update equations for the electric field.
- `Cb` , `Db` : Constants used in the update equations for the magnetic fields.

Main Logic

The core of the simulation is encapsulated in a loop that iterates over the specified number of timesteps. Here's a breakdown of the operations performed in each iteration:

1. If it is the first timestep, the central point of the electric field array `Ez` is initialized to 1, simulating an impulse.
2. The magnetic field along the y-axis (`Hy`) is updated based on the differences in the electric field.
3. The magnetic field along the x-axis (`Hx`) is updated similarly.
4. The electric field (`Ez`) is updated based on the current values of the magnetic fields.
5. The central point of the electric field is reset to 1.
6. Every fifth timestep, an image of the current state of the electric field is captured for animation.

Visualization

After the loop, an animation of the electric field is created using Matplotlib's `ArtistAnimation` feature. This animation displays the evolution of the electric field over time, providing a visual representation of the wave propagation.

Example Usage

To run the script, simply execute it in a Python environment where NumPy and Matplotlib are installed. You can observe the animated output showing the propagation of electromagnetic waves.

Conclusion

This script is a great example of how to implement FDTD methods for electromagnetic simulations. It's a useful starting point for anyone interested in computational physics and can be easily modified to explore different parameters and configurations.